QuakeSim Portal Software Test Plan

Contact: Marlon Pierce marpierc@indiana.edu 812-856-1212

Table of Contents

1. Scope

The project, “Numerical Simulations for Active Tectonic Processes: Increasing Interoperability and Performance” has the following primary aims:

- To develop and improve the performance of applications for the simulation of earthquakes and fault systems.
- To develop and deploy federated databases for storing and retrieving both real and simulated data.
- To develop a Web Services-based interoperability structure that can be used to access applications and bind them to data sources and user interface components.
- To develop an extensible portal system that can be used to aggregate and manage user interfaces to applications, data, and miscellaneous services.

Detailed descriptions and documentation of the project may be found at http://www-aig.jpl.nasa.gov/public/dus/quakesim/milestones.html.

This document covers the software test plan for code developed as part of Project Milestone I, "Interoperability". The foci of this milestone are the third and forth bullets above.

1.1 System Overview

The SERVOGrid framework provides the interoperability that binds the QuakeSim project's codes and data sources. SERVOGrid consists of collection of Web services for common tasks that we bind into Application Metadata Web Services. The user interfaces for accessing these services are delivered through a browser-based Web Portal. Component interfaces are managed as "portlets" within a portal container framework. Project details may be found at http://www-aig.jpl.nasa.gov/public/dus/quakesim/milestones.html and http://www.servogrid.org.

2. Referenced Documents

The following are supplemental documents that are needed to understand the test plan described herein:

- Software Development Plan: http://www-aig.jpl.nasa.gov/public/dus/quakesim/quakesim_sw_plan20020730.pdf.
- Requirements Document: http://www-aig.jpl.nasa.gov/public/dus/quakesim/CT_Requirements.doc.
- Software Design Document: http://www-aig.jpl.nasa.gov/public/dus/quakesim/DesignDocument2.1.doc.
- Portal Example User Manual: http://www-aig.jpl.nasa.gov/public/dus/quakesim/PortalExample.doc.

3. Software Test Environment

Software Version: The system describes QuakeSim portal software v 0.866025.

Languages/Compilers: All software described is written in Java. All machines use the Java 2 Software Development Kit version 1.4 for compiling and running software.

Third Party Software: All Web services are run by Tomcat 4.0/4.1/5.0 Web servers (http://jakarta.apache.org/tomcat/). All Web Services use Apache Axis versions 1.0 and 1.1 (http://ws.apache.org/axis/). The portal system base is built with Jakarta Jetspeed 1.4b4 (http://jakarta.apache.org/jetspeed/site/index.html). We use Apache Ant 1.6.x (http://ant.apache.org/) for both source compilation and for runtime execution management. These software packages are in turn built on several other projects; a complete list is available from the provided URLs. All third party software is freely available at no cost and open source.

Input Data Sets: Input data sets are provided. The test portal uses GeoFEST, Disloc, and Simplex as test applications. GeoFEST input data is generated in the portal from fault and layer data in the Fault Database. Sample Simplex and Disloc input data files were provided by developers.

Participating Organizations and Personnel: Choonhan Youn (IU), Galip Aydin and Marlon Pierce (IU) developed the Web service and portal software. Application codes were provided by Jay Parker (JPL), John Rundle and group (UC Davis), Robert Granat (JPL), and Terry Tullis (Brown). Each also provided assistance, documentation, and sample codes. Peggy Li (JPL) provided information and support for using RIVA, ParVox, and various driver scripts that were used for visualization. Anne Chen (USC) developed a Web service for accessing the fault database and provided client SQL query guidelines. Parker, Li, Theresa Baker (MIT/JPL) and Gerry Simila (Cal State Northridge) served as test users.

Testbed Architecture and Hardware: The portal system test environment consists of the following host machines:

1. complexity.ucs.indiana.edu: an 8 processor Sun Sunfire server. Complexity hosts the Web portal, identified as “User Interface Server” in Figure 1.
2. grids.ucs.indiana.edu: a dual processor Sun Ultra 60 server. Provides computational power for running science applications.
3. {danube,kamet,gridfarm008}.ucs.indiana.edu: dual-processor Linux servers. These machines are used to provide computational power for running science applications.
4. jabba.jpl.nasa.gov: An 8 processor SGI maintained by Jet Propulsion Laboratory with specialized visualization software (RIVA).
5. infogroup.usc.edu: A Linux server maintained by USC that hosts the Fault database.
6. orion.jpl.nasa.gov: 64 processor SGI Altrix used for ParVox demonstration.
7. losangeles.jpl.nasa.gov: Dual processor Linux server used to demonstrate adaptive meshing capability.

Unless otherwise specified, the testbed servers are owned and maintained by the Community Grids Lab.

The general architecture is depicted in the Figure 1:

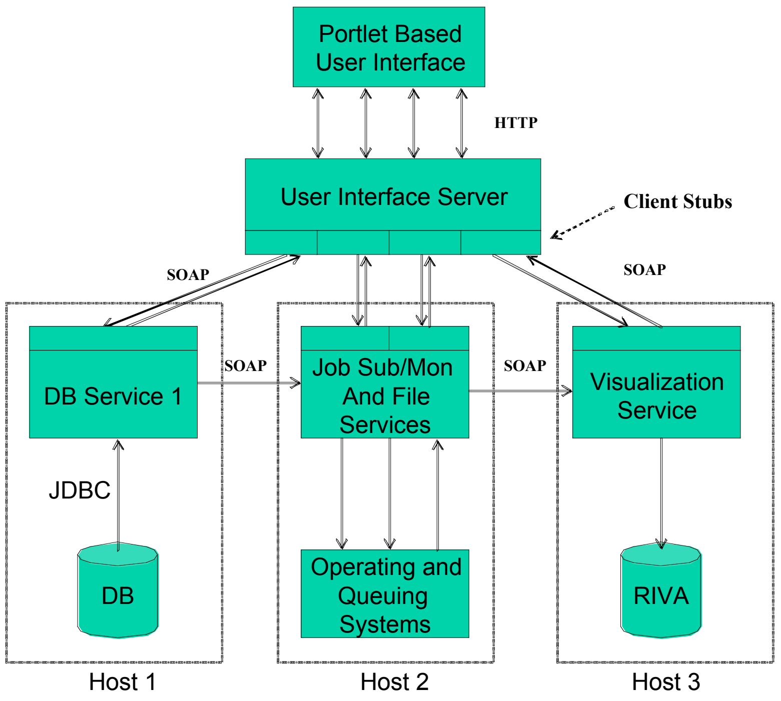

Figure 1 QuakeSim portal architecture

The DB service is located on infogroup.usc.edu. The visualization service runs on jabba.jpl.nasa.gov. The User Interface Server is hosted on complexity. All other listed hosts provide Job Submit/Monitor services. In the portal walkthrough described in the User Manual, danube acts as the compute host.

Configuring the System: The User Interface (UI) server and all backend service hosts shown in Figure 1 run Tomcat Web servers. The service hosts run Apache Axis as a web application in Tomcat. The UI server runs Apache Axis and Jetspeed as web applications. These may be configured in the standard ways, as described in the URLs above.

All system Web services that we developed may be deployed using standard methods described in the Apache Axis documentation. Clients stubs for these services may also by generated using the documentation described therein. Jetspeed portal extensions may be compiled and deployed using Apache Ant build scripts. User interface components are developed using JavaServer Pages and are deployed as a separate web application in the UI server.

4. Test Identification: General Information

All portal interface testing uses HTTPUnit, freely available from http://httpunit.sourceforge.net/. Unit tests are executed using through Apache Ant. We have a test suite to test the following for each code:

1. Navigation through the form process needed to submit the application on a selected host.
2. Loading sessions.
3. Downloading selected results.

Through successful interaction with the user interface, these automated tests verify several underlying services:

1. Portal login works correctly
2. WebFormPortlets successfully load sequences of interfaces.
3. Application metadata is correctly retrieved from the Application Metadata web service to construct application interfaces.
4. Context Metadata is correctly retrieved from the Context Manager web service to load archived sessions and their parameters.
5. Job submission works correctly
6. File transfers/downloads work correctly.
7. Database access works correctly (for mesh generator, GeoFEST, and RDAHMM).

Executing the test suite on a regular basis also services as system testing to identify problems with remote service implementations: failed tests indicate remote servers need to be restarted.

5. HTTP Unit Tests

When every action happens, the success condition looks for the indicated text in the generated HTML page. If this text is missing, this indicates that a failure has occurred.

Test Login

Action	Success Condition (HTML Text)
Enter username and password and click submit button	Code Selection Menu

Test Disloc

1. MainMenu

Action	Success Condition (HTML Text)
Click Disloc hyperlink	grids.ucs.indiana.edu
Set radio button and click Make Selection button	Disloc is a code for simulating surface displacements for a given fault geometry model.

2. New Project

Action	Success Condition (HTML Text)
Click New Project button	GEM Disloc Input: Observation Points
Click Make Selection button	GEM Disloc Input: Fault Model
Click Make Selection button	Your job was submitted. You can check the status using the Job Monitor.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	GEM Disloc Input: Observation Points
Click Make Selection button	GEM Disloc Input: Fault Model
Click Make Selection button	Your job was submitted. You can check the status using the Job Monitor.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	session1065379785585.out

Test Simplex

1. MainMenu

Action	Success Condition (HTML Text)
Click Simplex hyperlink	grids.ucs.indiana.edu
Set radio button and click Make Selection button	Simplex

2. New Project

Action	Success Condition (HTML Text)
Click new project button	GEM Simplex Data service
Click Make Selection button	Fault Data
Click Make Selection button	Your job was submitted. You can check the status using the Job Monitor.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	GEM Simplex Data service
Click Make Selection button	Fault Data
Click Make Selection button	Your job was submitted. You can check the status using the Job Monitor.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	session1082658104406GEM output

Test GeoFEST_Plus_Viz

1. MainMenu

Action	Success Condition (HTML Text)
Click GeoFEST_Plus_Viz hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	Geometry Creation and Mesh Generation

2. Delete Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Click Make Selection button	GeoVizTest
Set GeoVizTest radio button and Click Delete button	Not find “GeoVizTest”

3. New Project

Action	Success Condition (HTML Text)
Click new project button	Project Input
Click Make Selection button	Project Name: GeoVizTest
Set create new layer radio button and Click Make Selection button	Input Solid Layer Geometry
Click Selection button	mylayer
Set create new fault radio button and Click Make Selection button	Input Fault Geometry
Click Selection button	myfault
Set load layer radio button and Click Make Selection button	LayerName
Set NorthridgeAreaMidCrust layer radio button and Click SelectLayDBEntry button	Input Solid Layer Geometry
Click Selection button	NorthridgeAreaMidCrust
Click Generate Mesh button	Refine Mesh
Click Refine Mesh button	TETRAHEDRA
Click Save Mesh button	Launch GeoFEST
Click Launch GeoFEST button	Your job has been launched on danube.ucs.indiana.edu.

4. Load Project

Action	Success Condition (HTML Text)
Click load project button	GeoVizTest
	Current Project Components

Set GeoVizTest radio button and Click Select button	myfault
	mylayer
	NorthridgeAreaMantle
	NorthridgeAreaMidCrust
Click Generate Mesh button	Refine Mesh
Click Refine Mesh button	TETRAHEDRA
Click Save Mesh button	Launch GeoFEST
Click Launch GeoFEST button	Your job has been launched on danube.ucs.indiana.edu.

5. Fetch Mesh

Action	**Success Condition (HTML Text)**
Click Fetch Mesh button	GeoVizTest
	Tetra

Test MeshGenerator

1. MainMenu

Action	Success Condition (HTML Text)
Click MeshGenerator hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	Geometry Creation and Mesh Generation

2. Delete Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Click Make Selection button	httpTest
Set GeoVizTest radio button and Click Delete button	Not find “httpTest”

3. New Project

Action	Success Condition (HTML Text)
Click new project button	Project Input
Click Make Selection button	Project Name: httpTest
Set create new layer radio button and Click Make Selection button	Input Solid Layer Geometry
Click Selection button	Mylayer
Set create new fault radio button and Click Make Selection button	Input Fault Geometry
Click Selection button	Myfault
Set load layer radio button and Click Make Selection button	LayerName
Set NorthridgeAreaMidCrust layer radio button and Click SelectLayDBEntry button	Input Solid Layer Geometry
Click Selection button	NorthridgeAreaMidCrust
Click Generate Mesh button	Refine Mesh
Click Refine Mesh button	TETRAHEDRA
Click Save Mesh button	Launch GeoFEST
Click Launch GeoFEST button	Your job has been launched on danube.ucs.indiana.edu.

4. Load Project

Action	Success Condition (HTML Text)
Click load project button	httpTest
Set GeoVizTest radio button and Click Select button	Current Project Components
	Myfault
	Mylayer
	NorthridgeAreaMantle

	NorthridgeAreaMidCrust
Click Generate Mesh button	Refine Mesh
Click Refine Mesh button	TETRAHEDRA
Click Save Mesh button	Launch GeoFEST
Click Launch GeoFEST button	Your job has been launched on danube.ucs.indiana.edu.

5. Fetch Mesh

Action	**Success Condition (HTML Text)**
Click Fetch Mesh button	httpTest
	Tetra

Test Geofit

1. MainMenu

Action	Success Condition (HTML Text)
Click Geofit hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	Geofit

2. New Project

Action	Success Condition (HTML Text)
Click new project button	Paste input field script here.
Set text fields and Click run code button	These pages will guide you through the steps needed to run Geofit to simulate faults.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	Paste input field script here.
Click run code button	These pages will guide you through the steps needed to run Geofit to simulate faults.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	testGeofit
Click GMT Image button	This page is best viewed with Internet Explorer.
	pdf

Test RDAHMM

1. MainMenu

Action	Success Condition (HTML Text)
Click RDAHMM hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	RDAHMM

2. New Project

Action	Success Condition (HTML Text)
Click new project button	RDAHMM Input Forms
Set text fields and Click run code button	These pages will guide you through the steps needed to run RDAHMM to simulate faults.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	RDAHMM Input Forms
Click run code button	These pages will guide you through the steps needed to run RDAHMM to simulate faults.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	Plan5.A

Test Slider

1. MainMenu

Action	Success Condition (HTML Text)
Click Slider hyperlink	Kamet.ucs.indiana.edu
Set radio button and click Make Selection button	Slider

2. New Project

Action	Success Condition (HTML Text)
Click new project button	Slider Input Forms
Set text fields and Click run code button	These pages will guide you through the steps needed to run Slider to simulate faults.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	Slider Input Forms
Click run code button	These pages will guide you through the steps needed to run Slider to simulate faults.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	testSlider.stdout

Test PatternInformatics

1. MainMenu

Action	Success Condition (HTML Text)
Click PatternInformatics hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	PatternInformatics

2. New Project

Action	Success Condition (HTML Text)
Click new project button	PatternInformatics Input Forms
Set text fields and Click run code button	These pages will guide you through the steps needed to run PatternInformatics to simulate faults.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	PatternInformatics Input Forms
Click run code button	These pages will guide you through the steps needed to run PatternInformatics to simulate faults.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	deltaP.xyz
Click Plot button	This page is best viewed with Internet Explorer.

Test GeoFEST2

1. MainMenu

Action	Success Condition (HTML Text)
Click GeoFEST hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	These pages will guide you through the steps needed to run GeoFEST2 to simulate faults

2. Delete Project

Action	Success Condition (HTML Text)
Click Run GeoFEST2 button	Geometry Creation and Mesh Generation
Click load project button	Select Projects
Click Make Selection button	httpTest
Set GeoVizTest radio button and Click Delete button	Not find “httpTest”

3. New Project

Action	Success Condition (HTML Text)
Click Run GeoFEST button	Geometry Creation and Mesh Generation
Click new project button	Project Input
Click Make Selection button	Project Name: httpTest
Set create new layer radio button and Click Make Selection button	Input Solid Layer Geometry
Click Selection button	Mylayer
Set create new fault radio button and Click Make Selection button	Input Fault Geometry
Click Selection button	Myfault
Set load layer radio button and Click Make Selection button	LayerName
Set NorthridgeAreaMidCrust layer radio button and Click SelectLayDBEntry button	Input Solid Layer Geometry
Click Selection button	NorthridgeAreaMidCrust
Click Generate Mesh button	Refine Mesh
Click Refine Mesh button	TETRAHEDRA
Click Save Mesh button	Launch GeoFEST
Click Launch GeoFEST button	Your job has been launched on danube.ucs.indiana.edu.

4. Load Project

Action	Success Condition (HTML Text)
Click Run GeoFEST button	Geometry Creation and Mesh Generation

Click load project button	httpTest
Set GeoVizTest radio button and Click Select button	Current Project Components
	Myfault
	Mylayer
	NorthridgeAreaMantle
	NorthridgeAreaMidCrust
Click Generate Mesh button	Refine Mesh
Click Refine Mesh button	TETRAHEDRA
Click Save Mesh button	Launch GeoFEST
Click Launch GeoFEST button	Your job has been launched on danube.ucs.indiana.edu.

5. Fetch Mesh

Action	**Success Condition (HTML Text)**
Click Run GeoFEST button	Geometry Creation and Mesh Generation
Click Fetch Mesh button	httpTest
	Tetra

6. Archived Data

Action	**Success Condition (HTML Text)**
Click Archived Data button	QUAKES1.out

Test GeneticAlgorithm

1. MainMenu

Action	Success Condition (HTML Text)
Click GeneticAlgorithm hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	GeneticAlgorithm

2. New Project

Action	Success Condition (HTML Text)
Click new project button	GeneticAlgorithm Input Forms
Set text fields and Click run code button	These pages will guide you through the steps needed to run GeneticAlgorithm to simulate faults.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	GeneticAlgorithm Input Forms
Click run code button	These pages will guide you through the steps needed to run GeneticAlgorithm to simulate faults.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	testGeneticAlgorithm.output1

Test Karhunen_Loeve

1. MainMenu

Action	Success Condition (HTML Text)
Click Karhunen_Loeve hyperlink	danube.ucs.indiana.edu
Set radio button and click Make Selection button	Karhunen_Loeve

2. New Project

Action	Success Condition (HTML Text)
Click new project button	Karhunen_Loeve Input Forms
Set text fields and Click run code button	These pages will guide you through the steps needed to run Karhunen_Loeve to simulate faults.

3. Load Project

Action	Success Condition (HTML Text)
Click load project button	Select Projects
Set radio button and Click Select button	Karhunen_Loeve Input Forms
Click run code button	These pages will guide you through the steps needed to run Karhunen_Loeve to simulate faults.

4. Archived Data

Action	Success Condition (HTML Text)
Click Archived Data button	trimpcafile

6. Test Schedules

The above tests were or are being conducted according to the following schedule.

7. Requirements Traceability

See attached requirements traceability matrix.